AFRL-IF-RS-TR-2007-124
Final Technical Report
April 2007

TOOL INTEGRATION FRAMEWORK FOR BIO-INFORMATICS

Vanderbilt University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. T713

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-IF-RS-TR-2007-124 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
CLARE D. THIEM Work Unit Manager	JAMES A. COLLINS, Deputy Chief Advanced Computing Division Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
APR 2007

2. REPORT TYPE
Final

3. DATES COVERED *(From - To)*
Mar 05 – Oct 06

4. TITLE AND SUBTITLE
TOOL INTEGRATION FRAMEWORK FOR BIO-INFORMATICS

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-05-2-0097

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)
Sandeep Neema

5d. PROJECT NUMBER
BIOC

5e. TASK NUMBER
VA

5f. WORK UNIT NUMBER
ND

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Vanderbilt University
110 21st Ave S.
Nashville TN 37203-2416

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-124

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07-205

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Experience in developing tool integration solutions for engineer domains indicates that the key ingredient for robust tool integration is semantically precise definition of interchange format. The project described in this report developed a robust, scalable, extensible, and standards-based tool integration framework for systems biology applications. Model-based techniques were developed and applied to formally and precisely define the interchange formats using metamodels and semantic well-formedness rules. These interchange formats were used for integrating diverse systems biology computational tools. The report describes the tool integration framework, interchange formats, and applications to representative systems biology workflows.

15. SUBJECT TERMS
Tool, integration, embedded system, toolchains, quality control repository, systems biology

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Clare Thiem
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER (*Include area code)***
U	U	U	UL	29	

Table of Contents

List of Figures

1. Introduction

Tool Integration is a challenging problem in a wide variety of domains. Systems Biology is one such domain, where biologists apply computational techniques to understand, and sometimes manipulate biological systems. Often these computational techniques are implemented in different tools, with different formulations of the problem. In order to efficiently and effectively, make these powerful computational techniques amenable and available to biologists, a *robust tool integration framework* that alleviates the systems engineering and integration problems is desirable. The lack of robustness, scalability, extensibility, and standards-base in the tool integration framework amplify systems engineering and integration costs, and the consequential delays hamper the overall growth of the nascent yet promising field of systems biology.

The state-of-the-art of integration frameworks in systems biology is BioSPICE Dashboard [5], a Java NetBeans [11] based Integrated Development Environment (IDE) for developing modules and packaging computational tools. The framework is extremely powerful in integrating computational tools as plug-ins and facilitates movement of data from one tool to another. A significant limitation however, is the lack of formalization in interchange formats. The interchange formats are declared as MIME-types which does not allow representation of semantics. This results in a constant reengineering as there is no explicit agreement on the meaning of the data being interchanged. Tool developers subject the interchange format to their own interpretation, and the overall integration produces misleading or erroneous results.

The goal of this project therefore, is to maturate the existing tool integration infrastructure, by leveraging experiences gained in developing tool integration solutions for engineering domains, and using advanced software engineering principles and techniques such as Model-based Design [1]. A key insight gained from developing tool integration solutions for engineering domains is that tool integration is more than just the act of moving data and models from one tool to another. Fundamental to a tool is the semantics of the data that it processes, manipulates, or generates. The soundness of tool integration is derived from semantically precise understanding of a tool's data, and semantically correct transformation of one tool's data into another tool's data.

This project has created formal and precisely defined interchange formats, leveraging standards based techniques, and tools developed in prior/related government sponsored programs such as DARPA Model Based Integration of Embedded Systems (MoBIES). The interchange formats are described using Universal Modeling Language (UML) [16] Metamodels and the Object Constraint Language (OCL) [17] which are sound and succinct in capturing abstract syntax, and static semantics of interchange formats. The performer organization, Institute for Software Integrated Systems (ISIS), has developed a Model-based design infrastructure, and a collection of meta-programmable tools, that are used in the project: Generic Modeling Environment (GME) [8] is used to graphically specify the metamodels; Universal Data Model (UDM) [15] provides the ability to automatically generate programming API-s from the metamodel, allowing tool developers to: conveniently and efficiently manipulate the interchange format, and program at a high-level of abstraction; Graph Rewriting Engine and Transformation (GREaT) [9] tool is used for implementing complex interchange format transformations; and Open Tool Integration Framework (OTIF) [2],[12] is used to deploy complex tool integration scenarios.

The project also developed a technical repository based upon the ESCHER [7] model of a quality-controlled repository. The repository, in addition to being an active archive of software tools and interchange format, also offers services such as automated updates, quality guidelines, and quality assessment and review for software tools, support tools for individual developers, and special interest groups for bug and issues reporting tracking, and fixing.

In order to illustrate the use and benefit of robust tool integration, the project has created prototype tool integration use-cases using OTIF, and formal interchange formats. The interchange formats and the prototype use-cases have been disseminated via the ESCHER quality controlled repository at http://www.escherinstitute.org.

2. Project Overview

This project followed a four pronged strategy to maturate the tool integration infrastructure:

1. Organized activities in the BioCOMP program that led towards identification and formalizing interchange formats
2. Adapted and evolved the ISIS-developed infrastructure to better serve the needs of the Systems Biology domain
3. Selectively and incrementally apply the formalization, and the infrastructure to demonstrate the potential benefits of standards based technology, and serve unfulfilled tool integration needs in high-value use-cases
4. Created a quality-controlled repository following the ESCHER model to archive and disseminate systems biology tools

Robust definition of interchange formats as metamodels required community participation for understanding and defining semantics of the interchange formats. A working group of BioCOMP researchers, called the Interface Definition and Interoperability (IFDEF) working group, was formed, supported and co-lead by Institute for Software Integrated Systems (ISIS) [10], to formally specify the interchange formats. The group met alongside PI meetings and BioSPICE Hackathon meetings, and developed an inventory of the Systems Biology Markup Language (SBML) [13] variants and their interrelationships. This inventory of SBML variants was formalized and integrated into an SBML-extension metamodel. The working group also identified and formalized other interchange formats (see Results section).

2.1 Open Tool Integration Framework (OTIF)

The project built its tool integration foundations upon OTIF – a model-based open tool integration framework, developed at ISIS in prior DARPA funded research. OTIF was originally developed to address the following specific tool integration requirements:

- *It shall clearly separate syntactic, semantic, and control issues in tool integration.*
  When different tools are integrated, there are at least three different aspects to be considered: syntax, i.e. how to handle the syntactical differences among tools, semantics, i.e. how to handle semantic differences among tools, and control, i.e. how to handle the differences in

the control and interactions among tools. The framework should provide mechanisms for solving all these issues in a non-interfering manner (to the extent possible).

- *It shall be able to integrate tools without modifying them, if that is not feasible.*
  Tools have typically three interfaces that can be used for integration purposes: persistence interface (e.g. file import, export), API (e.g. direct COM-based API to access the tool's internals), and the GUI (e.g. an interceptor mechanism that "taps into" the event stream and drawing commands between the main tool component and its visual front-end). The framework should be able to work with any of these.
- *It shall support integration of tools that are deployed as web-services.*
  As a new trend in software deployment, expensive tools are often provided as a service accessible and usable via the web (and not as a directly downloadable and installable package). The framework should be able to naturally integrate these tools.
- *It shall support transforming the product of one tool into the input of another tool.*
  Pipelining data from tool to tool does the simplest kind of tool integration. Because of syntactical and semantic differences, the pipeline frequently involves transformations. However, transformations could also incorporate other operations than strict "rewriting": for example, merging. For instance, data produced by tool A and tool B must be merged to serve as an input to tool C. The framework should allow the organization of arbitrary transformations (1-to-1 and many-to-1).
- *It shall support simple techniques for simple translation needs.*
  Frequently, translations are trivial textual rewriting the input data into the output data. A number of techniques are available to solve these translation problems, e.g. search and replace using regular expressions; transformation of XML files using XSLT scripts, etc. The framework should allow the implementation of simple transformations using the available tools.
- *It shall support batch, transaction-oriented, and notification-based integration.*
  There are number of different strategies for managing the control across different tools. In a batch-based approach, a producer tool produces a dataset, which is then passed along to the next tool in the chain. In a transaction-oriented approach, a producer tool executes a "write" transaction on a shared database, which will result in changes on that database, and a consumer tool should execute a "read" transaction on that database to retrieve the data. In a notification-based approach, fine-grain changes are performed by a producer tool, which then sends notification messages to consumer tools (who subscribe to these notifications), which then perform appropriate incremental changes on their own. The framework should be able to support any and all of these techniques.

Based on these requirements, the following architecture was developed for OTIF (see Figure 1 below).

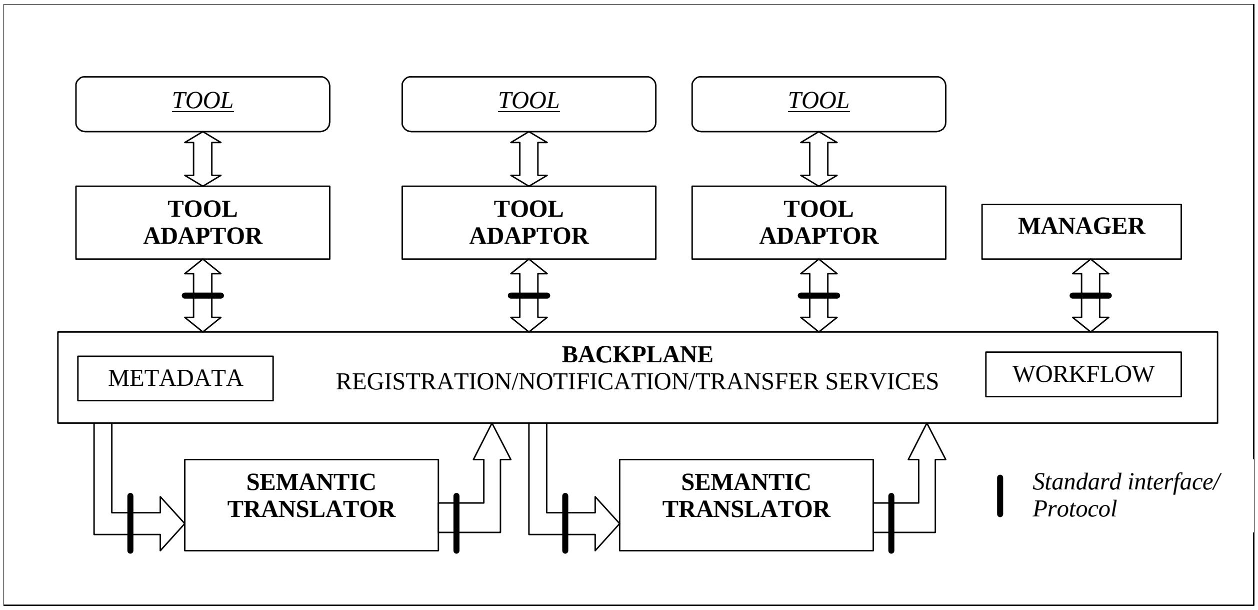


Figure 1: The Open Tool Integration Framework Architecture

The architecture consists of the following components (in addition to the design tools to be integrated):

- Tool Adaptor: This component is responsible for realizing the interface with the tool (any of the methods mentioned above) and performing syntactical transformations on the tool's data. The tool adaptor should convert all data coming from the tool into a canonical form, and pass it along to the backplane. Similarly, data coming from the backplane should be converted by the tool adaptor into tool-specific physical data. In the case of notification-based integration, the same applies to events generated and consumed by the tools: the tool adaptor performs the syntactic conversion on the events. Tool adaptors may have state for to support stateful interactions between tools.
- Semantic translator: This component is responsible for performing the semantic translation on data (or events) among different tools. In the simplest case, it performs a mere data rewriting, but in more complex cases the translations could be quite sophisticated. The translators relate producer tool(s) to consumer tool(s), although the most general, many-to-many case is possibly very rare.
- Backplane: This component is the backbone of the integration framework. It provides coordination services between the other components. The services include: registration and identification of components, notification, and physical data transfer. The backplane is typically distributed across multiple machines.
    - Workflow: Workflow models are loaded into the backplane and are used to facilitate the data transfer between tools and translators. Specifically, the backplane uses these models how to route the data (i.e. the models) among the different components.
    - Metadata: The metadata comprises the metamodels of all the tools registered with the backplane. These metamodels are loaded at initialization time and used for consistent labeling of model elements across the tools.

- Manager: This is a utility component for administration and debugging purposes. Administration involves enabling and disabling tools and users, etc., debugging operations allow run-time monitoring and troubleshooting the backplane.

The most challenging component in the above schema is the semantic translator. The semantic translator realizes the connection: the conceptual bridge between two (or many) tools. However, all semantic translations should operate in a common framework. This common framework could be grounded in the abstract syntax of the tools to be integrated. The abstract syntax defines what concepts a tool works with, what association exists among those concepts, what attributes belong to those concepts and associations, and what integrity constraints exist among the concepts and associations. Tool data should always comply with the abstract syntax of the tool. The semantic translation problem is approached by expressing it in terms of rewriting between two (or many) abstract syntax trees. On the “lowest level”, the translators are transforming data compliant with one abstract syntax definition into data compliant with another abstract syntax. Another view of semantic translation is that of transformations between type systems: the data is always typed and the translation can be defined between –perhaps quite complex— mappings. For the actual implementation of the semantic translators, the GReAT tool and framework is utilized.

The project extended the OTIF infrastructure to integrate an Eclipse front-end for Desktop Integration. Eclipse was chosen over Netbeans owing to a higher acceptance, better infrastructure architecture, and better tool support. OTIF standalone is a distributed tool integration framework, with a backplane component that stages the workflow and tool adapters that integrates the tools. The desktop front-end provides the ability to deploy a desktop workflow natively within Eclipse, all the tool adapters can be instantiated concurrently in Eclipse, and OTIF backplane runs in the back providing the necessary data transformation, and data transfer operations. The project created an OTIF perspective, an OTIF tool view, and an OTIF events view within Eclipse to enable this integration. The GenericToolAdapter, an Active-X component was extended to service multiple clients for the OTIF BackPlane server.

2.2 Universal Data Model

The Universal Data Model (UDM) tool generates programmatic API-s automatically from metamodels. These API-s could be use to manipulate models/data adherent to a specific metamodel regardless of the underlying data-persistence format. This enables the programmer to develop computation programs without worrying about the myriad details of data persistence, and persistence formats such as XML files, or Databases, or Binary files, or ASCII text files. The API-s generated with the existing UDM infrastructure at ISIS was in C++ programming language. This posed a significant limitation towards the use of the tool, since Java was the primary programming language used by Systems Biology tools developer. The project developed a Java interface for the UDM package, to overcome this limitation. The Java interface consists of the following elements:

- UDM compiler/Java generator. The UDM compiler was augmented to generate Java source code and programming API-s for accessing the UDM objects. The generated code consists of class definitions for all the classes described in the UML models, similarly to the generated code in C++.

- UDM/Java interface libraries. These libraries provide the generic, Java implementation of core UDM classes, and link to the C++ implementation libraries. The linkage to the C++ implementation code was accomplished via the Java Native Interface libraries and tools.

The implementation is a mixture of generated Java code, handwritten Java code, and generated Java/C++ interface code. This approach reuses the core UDM libraries (hence only one version should be maintained), and also provides high performance (that could be better than a pure Java-based approach). Figure 2 below shows the architecture of the UDM Java Extension.

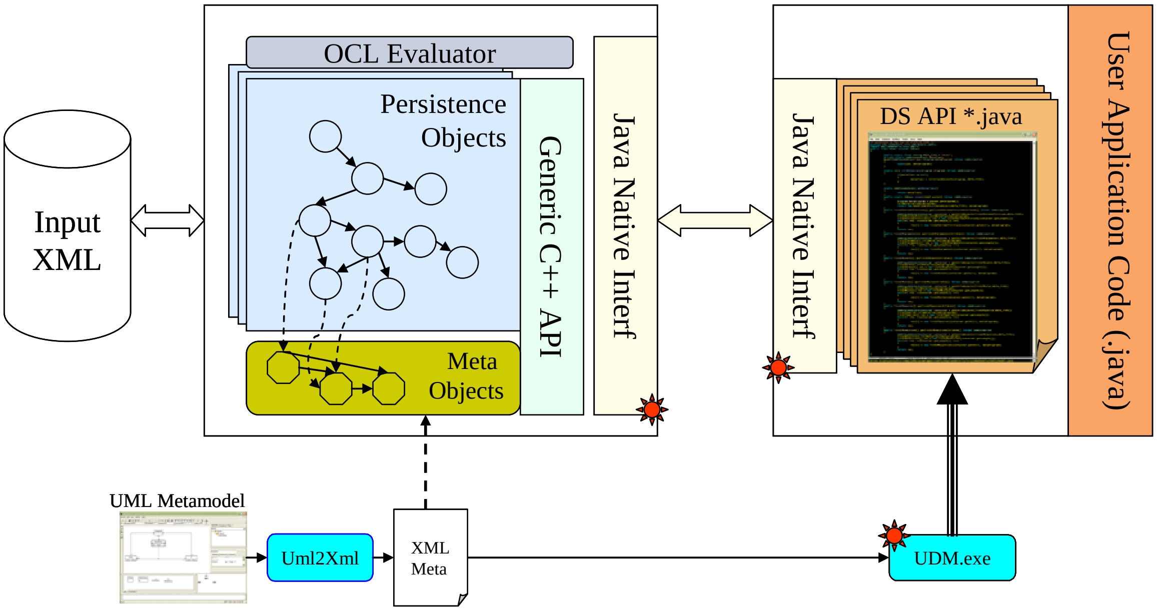


Figure 2: UDM Java Extension Architecture

2.3 ESCHER Quality Controlled Repository

The ESCHER Research Institute is a non-profit partnership between industry, government, and the research community. ESCHER offers a unique model for transitioning embedded computing technology and its mission is to provide value-added services that enable the transition of government-funded information technology research results to industry and government customers. This mission is accomplished via value-added activities focused on defining and managing research projects, maintaining and providing access to a repository of research products, and performing road mapping exercises that forecast the direction of new technologies and identify research needs for the government, military, and industry. ESCHER conducts activities as a non-profit corporation in order to operate as an independent organization that does not compete against commercial companies, universities, or research laboratories. This unique approach benefits the government and military, industry, and the research community.

The pace of technological innovation in key research areas has made the traditional models for transition inadequate. Accelerated product development cycles, globalization (e.g., global competition, global marketplaces, global labor pools), easy access to venture capital, heightened government responsiveness, and more system (versus component) innovations have all contributed to this. At the same time, there has been an added emphasis on consortia, partnering, and cost-sharing and an increased importance in “community development” and “open source” research and development efforts. Government agencies have identified embedded systems technology as a mission-critical area and have responded with a suite of research programs totaling more than $300M in recent years. ESCHER was organized and provided seed funding from DARPA and NSF in large part to ensure the fruits of such a large research investment are realized and to provide a vehicle for harvesting long-term, pre-competitive research results. ESCHER has introduced a new model for transition in rapidly evolving technology areas—an industry meeting place where corporate end-users can influence research sponsors and pose requirements specifications, challenge problems, and even full-blown testing environments to the community. The ESCHER model leverages pooled corporate funding to bring research systems the “final step” toward maturity and is a place where small companies may find new product opportunities and technologists will find customer-driven challenges to motivate their research and provide an eventual transition vehicle.

At the core of ESCHER is an operational and technical infrastructure designed to ensure the success of ESCHER’s unique model for transitioning embedded computing technology from government-funded research programs to military and commercial applications. A key component of that infrastructure is ESCHER’s technology repository. ESCHER provides access to its software services, tools, and frameworks (e.g., ACE, TAO, OTIF) through an online portal to its technology repository. This technology repository provides a description of each component, associated dependencies, supporting documentation, and a link to download the component. Access to certain components of the technology repository, however, are secure and require users to request and receive authorization for a repository account. The online portal provides links to automated updates, Really Simple Syndication (RSS) style news-feeds, developer support tools, special interest groups for bug and issue reporting, tracking, and fixing, and external websites and discussion groups. Additionally, the portal provides access to ESCHER’s Tool Qualification guidelines. These guidelines ensure that software and tools meet certain minimum criteria before being accepted into the technology repository and provide a means of communicating to contributors what is expected of their technology and letting users of the technology know what they’re getting. ESCHER’s Tool Qualification guidelines cover the following topics: Intellectual Property Rules, Tool Dependencies, Functional Integrity, Integrability, Documentation, User Support, Quality Assurance, and Domain Applicability.

3. Results

This section summarizes the project results. The Interchange format metamodels, the generated API-s, the extended OTIF are available for download from the ISIS BioCOMP website: http://www.isis.vanderbilt.edu/project/biocomp, together with some of the prototype toolchains that were constructed. These artifacts are also archived in the ESCHER quality controlled repository.

3.1 Application Domain Results: Interchange Formats

One of the major contributions of this project is formalized interchange formats described as metamodels. The key interchange formats are described below along with their metamodels.

3.1.1 SBML

Systems Biology Markup Language (SBML) is a community supported and widely used interchange format for exchanging systems biology models. SBML allows representation of biochemical reaction networks. One of the key challenges of SBML is that depending on the nature of the reaction network, different operational semantics can be ascribed to the SBML model. Examples of such variant semantics include continuous dynamics, hybrid dynamics, stochastic, and flux-balance dynamics. Such semantic variants or “dialects” present significant challenge for interoperability since mere expression of the fact that a tool accepts SBML input or generates SBML output is not a sufficient condition for integrability of the tool. Moreover, SBML offers extensibility with the use of annotations, which are tags that could be attached to any modeling element in SBML and capture additional information. The semantic variants of SBML use annotations as an informal way to create an overlay language on top of “base” SBML.

This project took an approach of rendering the dialects a first-class status, by extending SBML to allow capturing the dialect information. The explication formally captured the overlay constructs for each dialect of SBML. Figure 3 shows snippets of the extended SBML metamodel. The tree-browser view shows different SBML dialects i.e. bc or Bio-Charon which has a hybrid dynamics, flux balance which has stochastic dynamics. Figure 4 shows a further “drill-down” into the dialects. Thus, the metamodel captures formally the overlay languages. The explicit modeling of dialects also enabled specifying dialect-specific semantic constraints that could then be enforced with UDM and its auto-generated API-s.

The project also organized a community effort to develop a set of semantic well-formedness rules that could be incorporated in the metamodel as OCL constraints for ensuring the semantic validity and correctness of the model.

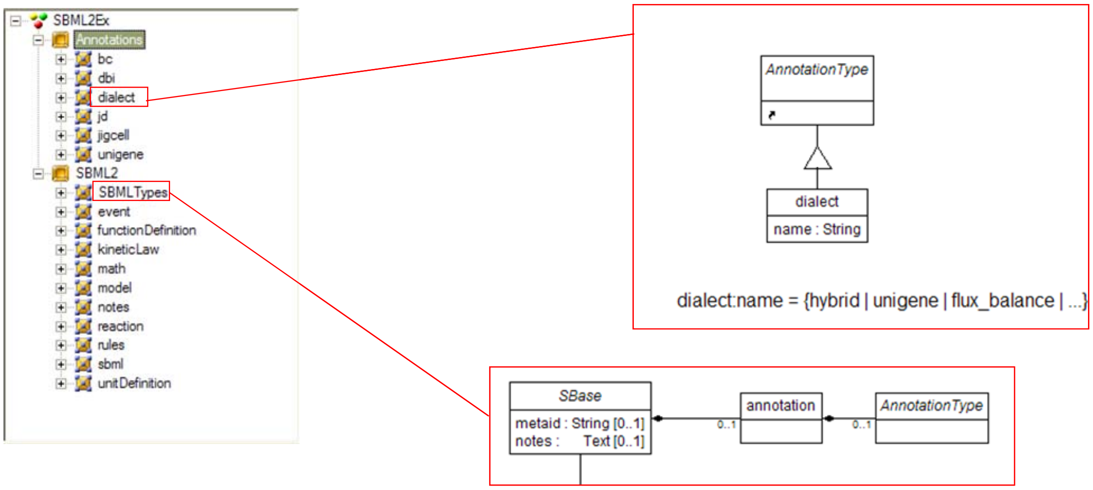


Figure 3: SBML2 Extension Metamodel (1)

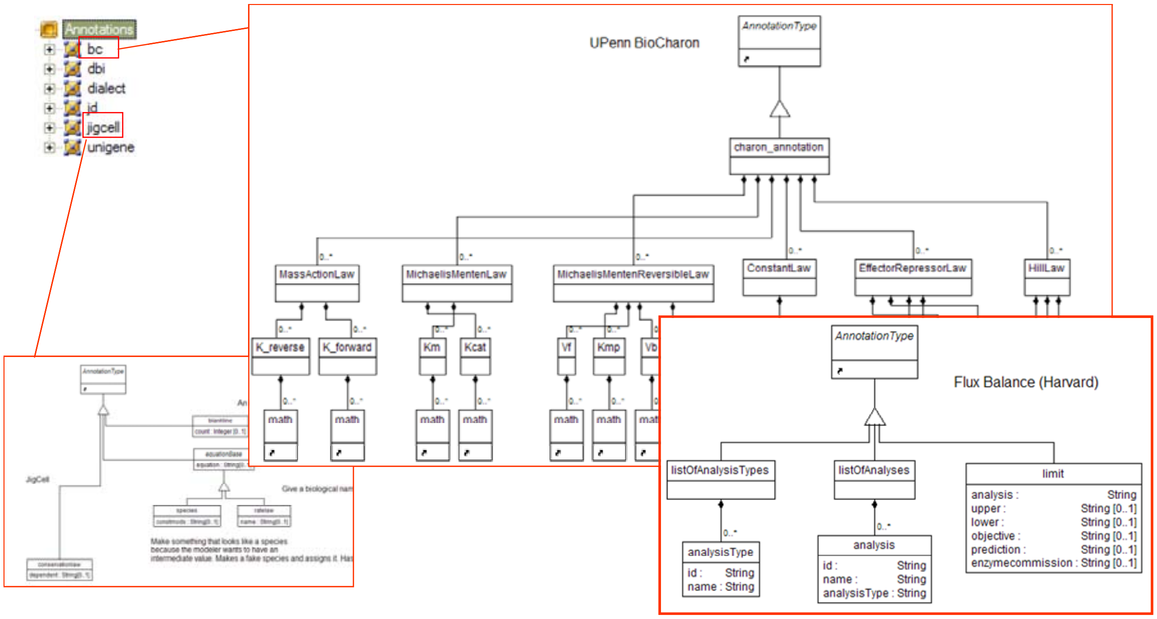


Figure 4: SBML2 Extension Metamodel (2)

3.1.2 GeneTF

The project developed in collaboration with researchers at Thomas Jefferson University, and Indiana University a data-type to capture Gene Transcription Factor Interaction. A major challenge in developing this data-type was to ensure the scalability, since the data-sets that the data-type is required to express are significantly large (~25000 genes, ~300 transcription factors).

Figure 5 shows the metamodel of GeneTF. The key concepts in metamodel are: Gene, TF (Transcription Factor), and an association element Regulation. The type attribute of Regulation characterizes the nature of interaction, whether it is up regulating, down regulating, or if the regulation is unknown. This simple yet precise description of the Gene Transcription Factor interaction network offers significant benefit in terms of semantic clarity and scalability. The UDM Java generated API-s are in community wide use.

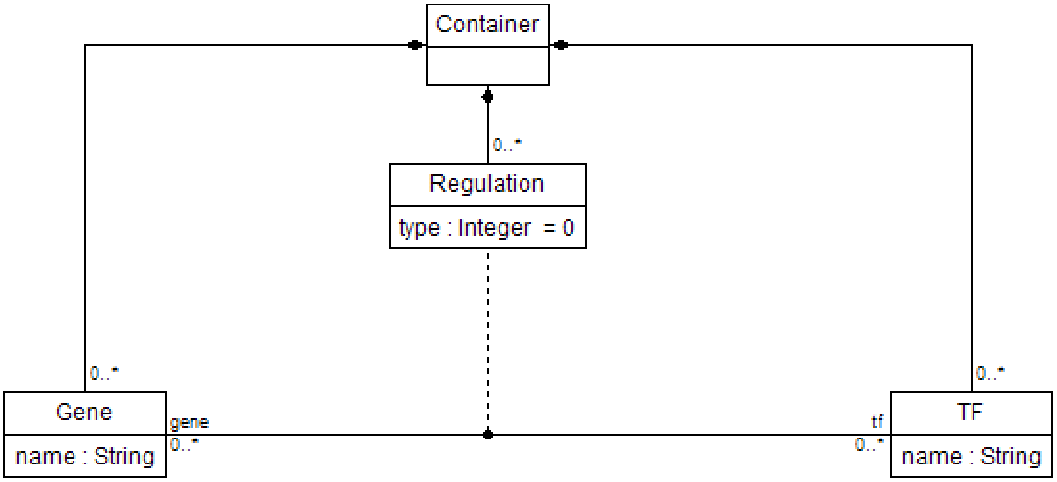


Figure 5: Gene Transcription Factor Interaction Network Metamodel

3.1.3 MageML

Microarray Gene Expression Markup Language (MAGE-ML) is a language designed to describe and communicate information about microarray based experiments. MAGE-ML is a large language with a large collection of constructs to describe microarray designs, microarray manufacturing information, microarray experiment setup and execution information, gene expression data and data analysis results.

Figure 6 shows snippets of the MAGE-ML metamodel. The metamodel is organized in several packages such as ArrayDesign_package, Array_package, BioSequence_package, Experiment_package, HigherLevelAnalysis_package etc., and composed and integrated into the MAGE-ML package. A set of semantic constraints have been developed and included in the metamodel that can be used to test the validity of the MAGE-ML model using the UDM generated API-s.

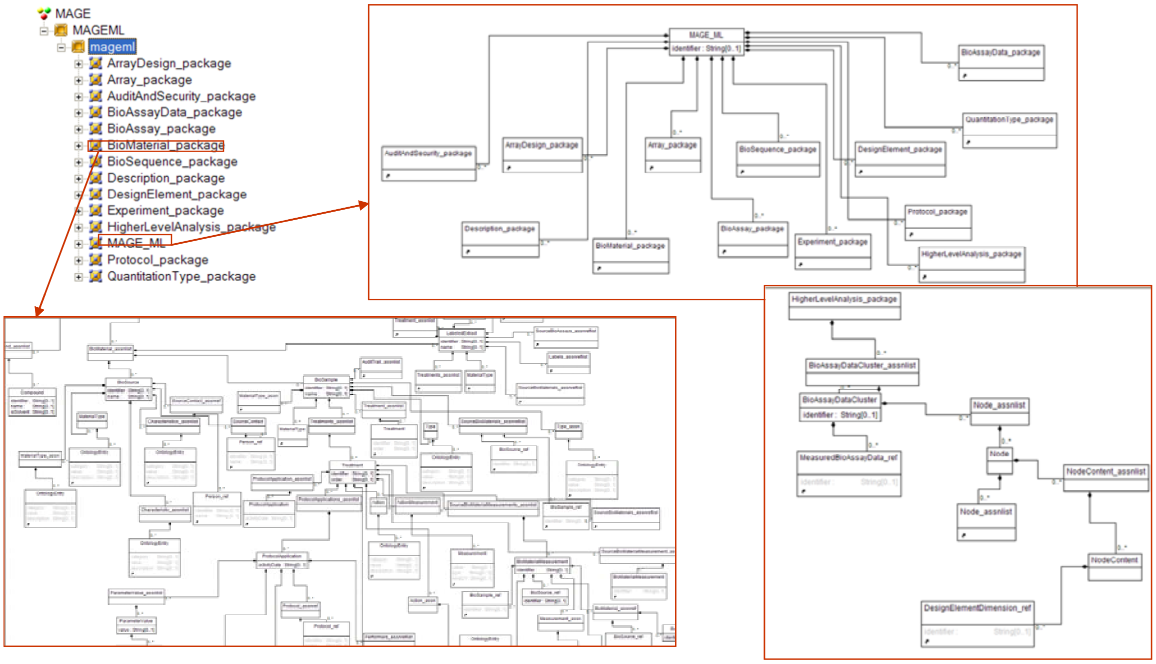


Figure 6: MAGE-ML Metamodel

3.1.4 TimeSeries

The TimeSeries is one of the basic interchange formats used by a variety of tools to accept or generate experiment data or computational results or observations. Figure 7 shows a metamodel of the timeseries interchange format. The key concepts and their relationship are depicted in the metamodel. The ‘Container’ contains a sequence of ‘time_point’ elements, which can be associated with 0 or more ‘data_value’ elements. ‘data_value’ is an abstract entity which can be one of ‘boolean_data’, ‘string_data’, ‘integer_data’ or ‘real_data’. A ‘data_label’ could be associated with a 0 or more ‘data_value’ elements.

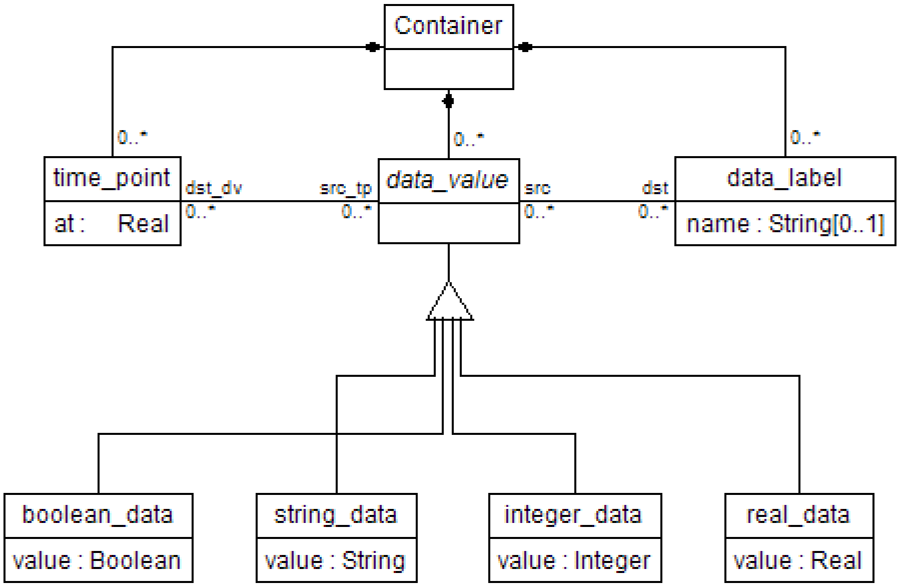


Figure 7: Timeseries Metamodel

3.2 Application Domain Results: Toolchains

The project created several integrated tool chains to be demonstrated as prototype use-cases of the tool integration technology. This section briefly demonstrates these integrated toolchains.

3.2.1 PAINT-NCA and NYU-PAINT toolchain

The project initially created proof-of-concept prototypes using the OTIF and the translator technology that were delivered and demonstrated. The prototypes are shown in Figure 8. The first integration solution — (a) on the figure — involved the Promoter Analysis and Interaction Network generation Tool (PAINT) [3][13] tool from Thomas Jefferson University and the Network Component Analysis (NCA) [4] tool from University of California Los Angeles. A translator that connected the two tools was created to form a simple toolchain. The PAINT tool generates its gene transcription factor association data as an SBML document, while the NCA tool expects the association information as a MathML document. Utilizing the SBML, and MathML meta-models listed above, a model transformer was developed using the graph-rewriting tool GReAT. This transformer was packaged as an XML wrapped analyzer for the Dashboard platform.

The toolchain was implemented using GReAT version r1.3.3, UDM version r2.21, PAINT version 3.2., NCA 0.9, and Dashboard version 5.0. This version of Dashboard ran with NetBeans IDE 3.6 requiring Java Runtime 1.4 on a machine with Windows XP. The toolchain is executed by loading a workflow into Dashboard, and running it.

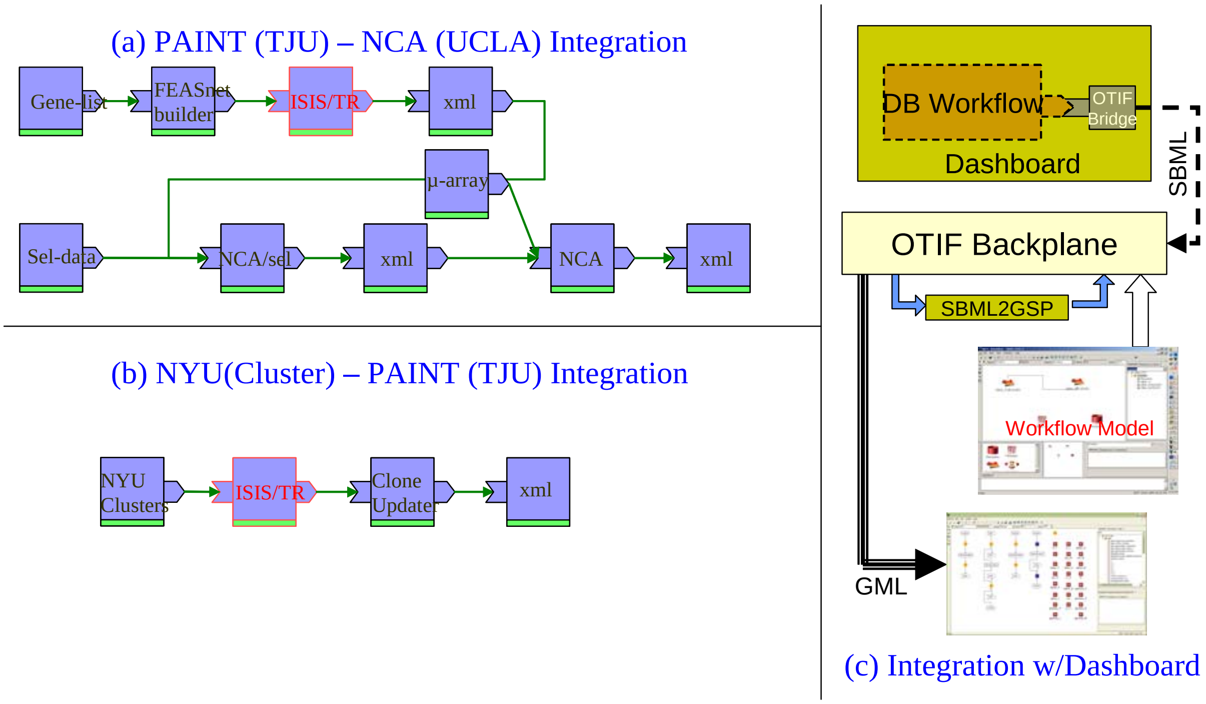


Figure 8: PAINT – NCA / NYU – PAINT toolchain

The second integration solution — (b) on the figure— included a gene clustering tool from New York University and the PAINT tool from TJU. Similarly to the previous one, we have built a semantics translator using the same technology as the one used in OTIF. For the (a) and (b) cases we have used the Dashboard infrastructure for deploying the toolchain for the reason that researchers on the program were already experienced with it (although the OTIF backplane, etc. could have been used as well).

The toolchain was implemented using GReAT version r1.3.3, UDM version r2.21, PAINT version 3.2., NYU-MAD vs. 1.0, and Dashboard version 5.0. This version of Dashboard ran with NetBeans IDE 3.6 requiring Java Runtime 1.4 on a machine with Windows XP. The toolchain is executed by loading a workflow into Dashboard, and running it.

As a third example — (c) on the figure—a bridge between OTIF and Dashboard was implemented, using the SBML (Systems Biology Modeling Language).

The project was thus able to facilitate integration between two major tools (NCA tool from UCLA, and PAINT tool from TJU) in the program. The results have been a major value addition in terms of the ability of the NCA tool to process much larger data sets being generated by biologists in the program

3.2.2 Model Estimation and Control with Particle Filter

The motivation for this use case was derived from the need to integrate wetlab experiments (bio-reactor) with high-performance modeling and simulation tools executing in a distributed computing environment in a real-time closed loop. Figure 9 shows a proposed architecture of model estimation and control. A BioReactor is a wetlab tool that can feeds nutrients to live cells and experimentally measures the metabolism. An interface to the BioReactor can enable it to post measurements in a time-series like format to the OTIF backplane periodically. OTIF notifies and feeds these measurements to the particle filter, which executes over cluster computing resources. The rationale to execute over the cluster computer is the computational complexity of the particle filter. The complexity is a function of the number of particles used for tracking. Usually a larger number of particles give a better resolution, though the exact tradeoff depends on the type of model - whether it is non-linear, whether it is hybrid etc. OTIF sends measurement updates to the Particle Filter. The Particle Filter -based on the latest measurements publishes its best hypothesis model to the OTIF backplane again. This can be in an SBML (or a SBML-like) format. In case there are differences from SBML it could be to transformed into SBML with a translator plugged into OTIF. The SBML file is dispatched to Dashboard, through a general-purpose (configurable file-formats) OTIF bridge module. The overall workflow within Dashboard includes three instances of OTIF bridge, a model simulator, and a model comparator. The simulator simulates the hypothesis model that is then compared with the Bio-Reactor generated experimental data to compute the confidence in the model. The confidence (model evaluation) results are fed back to the Particle Filter through the OTIF bridge.

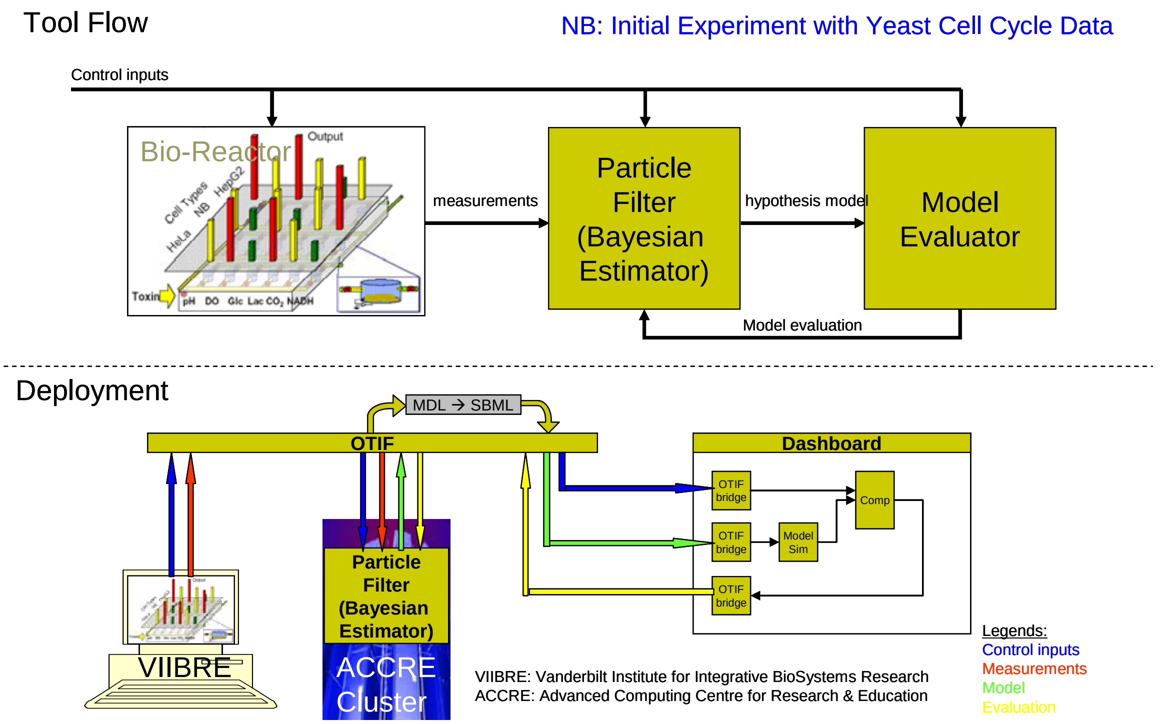


Figure 9: Model Estimation and Control with Particle Filter

Owing to limited time, resources, and experience with bio-reactor, all components of this use-case could not be realized. However, several crucial components of this workflow were realized.

A distributable particle filter estimator was developed, using the Parallel Virtual Machines (PVM) libraries for filter distributions. The particle filter was tested rigorously with data to establish the functional correctness of the design. The data sets for initial experimentation came from a hybrid systems problem - a 3-tank system that has both continuous and discrete dynamics. The particle filter was able to accurately and robustly track the 3-tank system's state space in the presence of noise. The number of particles was traded-off with accuracy in the experimentation. The results for the 3-tank system were fairly accurate with 800 particles. The computational complexity of the filter depended on the type of filter used in the particle filter. Two different update schemes were experimented with in the Particle Filter design: a) Kalman filter based, and b) Simple update. The Kalman filter was computationally intense, while the simple update (with a large number of particles) resulted in sufficiently accurate results. The final design choice therefore was to use a simple update equation in the filter, while tracking and estimating the state space with a large number of particles.

OTIF-Dashboard bridge modules were developed. Extension tasks were proposed to the Dashboard developers to augment its “run-to-completion” semantics – i.e. Dashboard starts from a source module and runs till the destination module has finished running, at which time it terminates the workflow. The proposed augmentation was to send an external signal to the Dashboard to rerun the workflow.

Figure 10 shows the Model Estimation and Control workflow as deployed. The BioReactor interface was replaced with a FileReader interface that produced time-series data similar to what the BioReactor would have produced. A MasterParticle Filter attached to the OTIF Backplane received these measurement updates and dispatched those to a number of Client ParticleFilters executing on a cluster computer. The output of the Particle Filter was send to the Dashboard via the OTIF-Dashboard bridge component. Within Dashboard the workflow included the OTIF-Dashboard Bridge, and a BioMat Bridge component. The BioMat Bridge is a community developed Dashboard component that can connect Dashboard to MATLAB. MATLAB was used to execute a different implementation of an Extended Kalman Filter tracking filter. The results of the Particle Filter were compared with those from the Extended Kalman Filter to establish the functional validity of the results from the Particle Filter.

The integrated workflow thus demonstrated an initial step towards integrating OTIF and Dashboard, and laid the initial path for developing heterogeneous tool integration solutions involving real-time closed loop interactions between in-vivo and in-silico tools.

This workflow was implemented with GReAT r1.5.1, UDM r3.0.1, OTIF v1.5.0, MATLAB r2006a, PVM 3.4.5, and Dashboard v 7.0. The workflows requires a PC with Windows XP to execute In order to run the workflow load start the OTIF Backplane server, upload the workflow to the Backplane server, start Dashboard and load the workflow, and start the Master Particle Filter application, which will then start pumping data to the workflow.

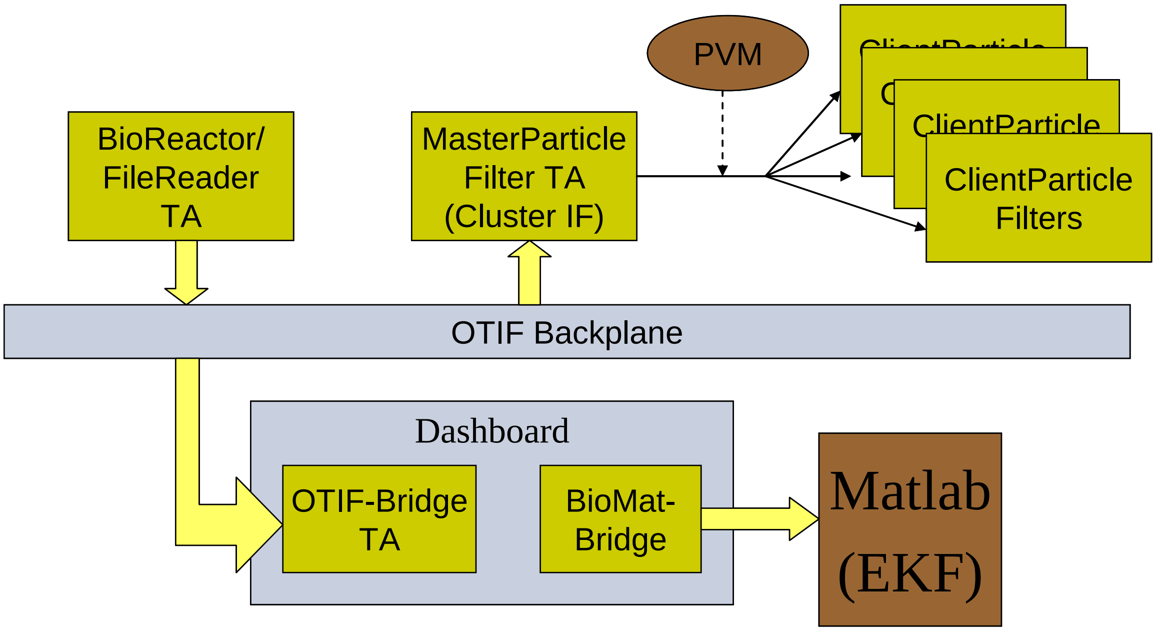


Figure 10: Particle Filter Toolchain as Deployed

3.2.3 Model Estimation with Machine Learning (Zebra Fish)

This use case was motivated by interest from Vanderbilt Medical School researchers, and it involves the study of the embryonic cell growth in Zebra Fish. The development of the vertebrate body plan is guided by a series of organizing events. The Speemann-Mangold organizer directs key morphogentic events during gastrulation like the formation of a dorsoventral axis. The genetic network that drives organizer formation and function is incompletely understood especially in Zebrafish. Zebrafish are emerging as a potent model organism to study human disease and the organizer is also a model for self organizing pattern formation. Understanding how the organizer directs pattern formation, can render it possible to co-opt this technology for nanofabrication of next generation self organizing devices. The immediate goal of the use case is to learn networks from data through simulation of known or presumptive assumptions. The long term goal is to develop formalized reasoning tools, based on model-data comparisons, to intelligently direct research. Figure 11 shows the notional workflow in the use case.

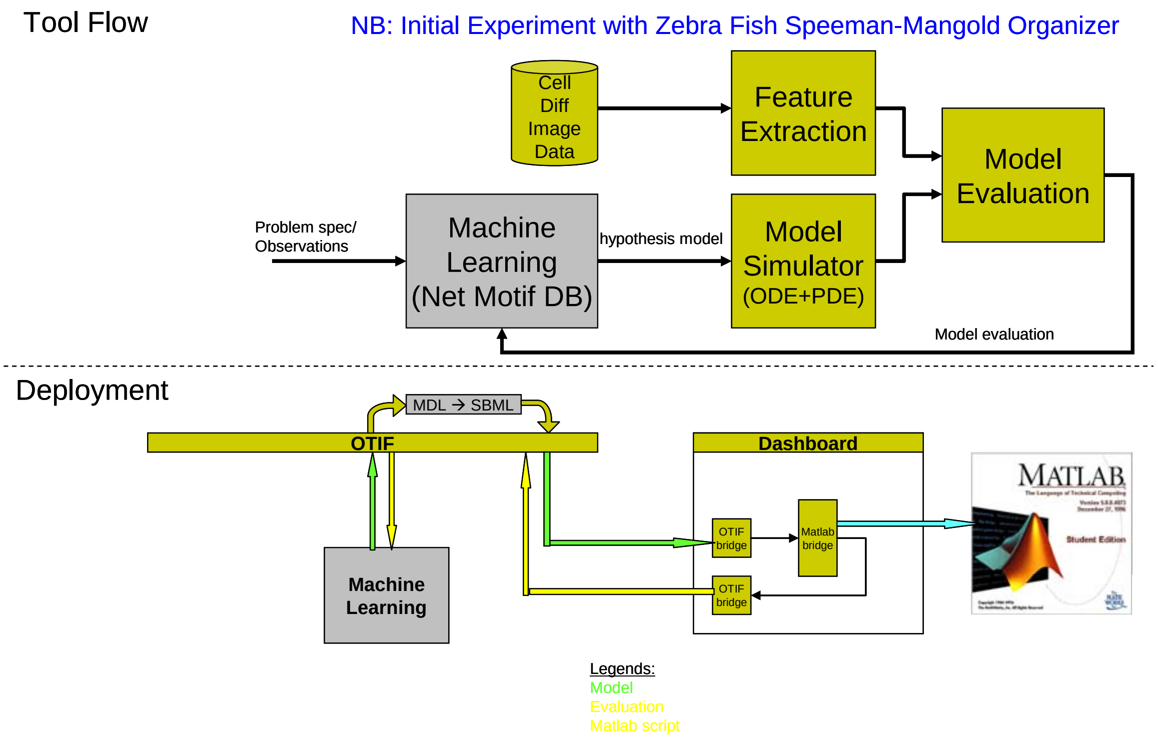


Figure 11: Model Estimation with Machine Learning (Zebra Fish)

Figure 12 below shows the experimental data gathered from Zebra Fish experiments [18]. The data shows the effect of initial concentration on developmental stages.

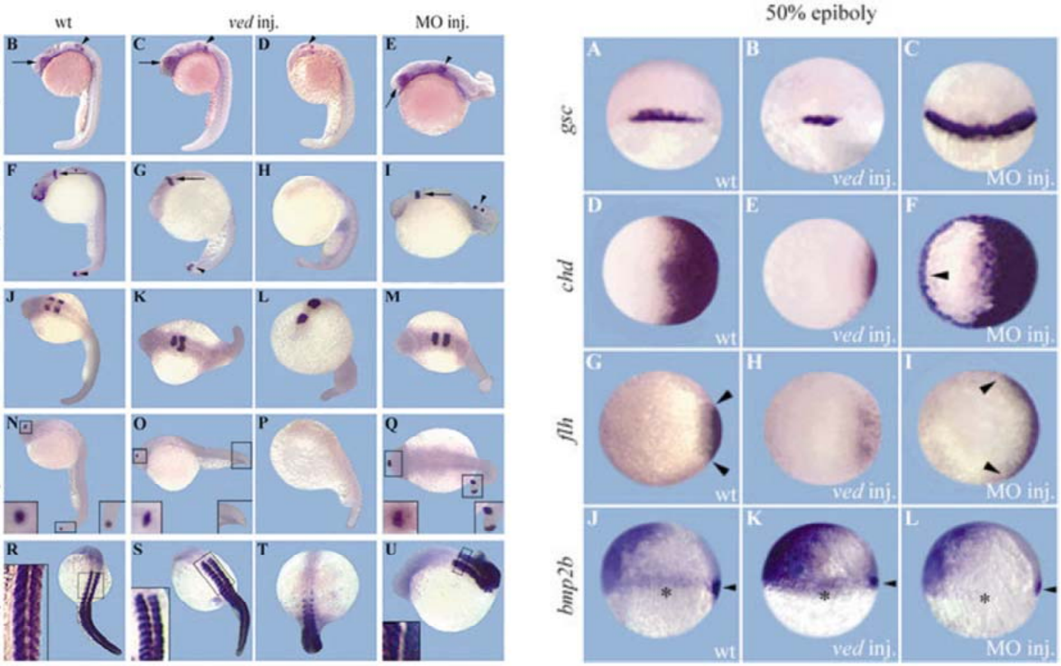
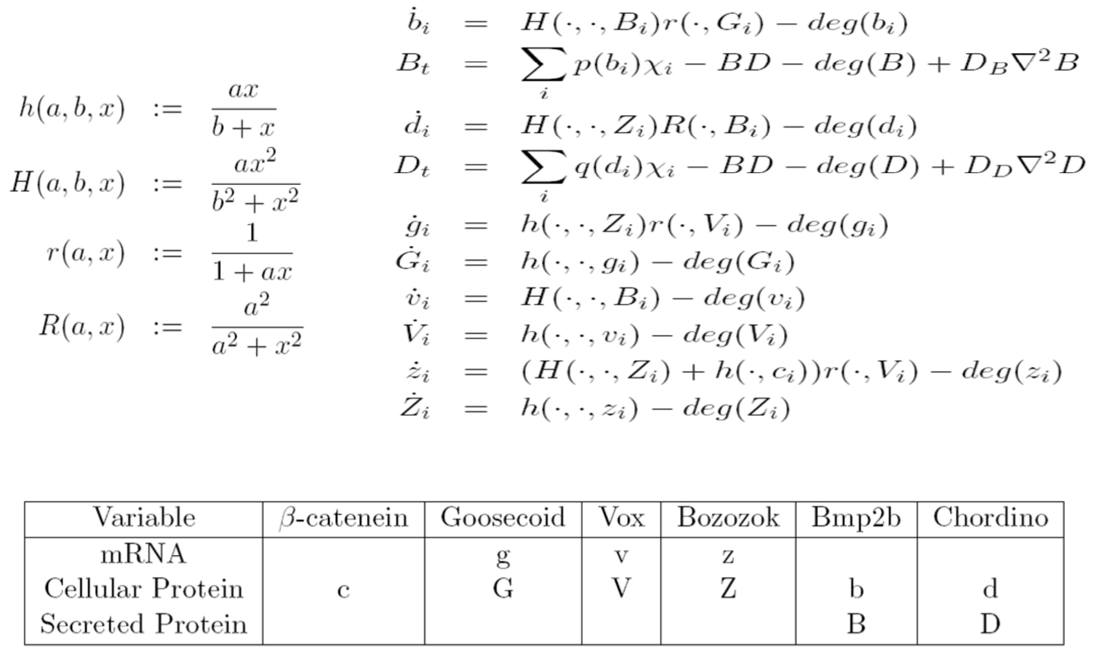


Figure 12: Zebra Fish Experimental Data

The Zebra Fish growth and development process could be characterized with two types of genes: a) Non-diffusable – there is no spatial component for these agents, they participate only in intra-cellular interactions, and can be represented with a single variable; and b) Diffusable – these genes have both a spatial and temporal component, participate in both inter and intra-cellular interactions, and are represented with two variables one for intra-cellular, and other one for inter-cellular. The production and consumption rates of these genes are captured with non-linear partial differential equations as shown in Figure 13 below. These equations were modeled and solved using MATLAB.



$$
\begin{aligned}
h(a,b,x) &:= \frac{ax}{b+x} \\
H(a,b,x) &:= \frac{ax^2}{b^2+x^2} \\
r(a,x) &:= \frac{1}{1+ax} \\
R(a,x) &:= \frac{a^2}{a^2+x^2}
\end{aligned}
$$

$$
\begin{aligned}
\dot{b}_i &= H(\cdot,\cdot,B_i)r(\cdot,G_i) - deg(b_i) \\
B_t &= \sum_i p(b_i)\chi_i - BD - deg(B) + D_B\nabla^2 B \\
\dot{d}_i &= H(\cdot,\cdot,Z_i)R(\cdot,B_i) - deg(d_i) \\
D_t &= \sum_i q(d_i)\chi_i - BD - deg(D) + D_D\nabla^2 D \\
\dot{g}_i &= h(\cdot,\cdot,Z_i)r(\cdot,V_i) - deg(g_i) \\
\dot{G}_i &= h(\cdot,\cdot,g_i) - deg(G_i) \\
\dot{v}_i &= H(\cdot,\cdot,B_i) - deg(v_i) \\
\dot{V}_i &= h(\cdot,\cdot,v_i) - deg(V_i) \\
\dot{z}_i &= (H(\cdot,\cdot,Z_i) + h(\cdot,c_i))r(\cdot,V_i) - deg(z_i) \\
\dot{Z}_i &= h(\cdot,\cdot,z_i) - deg(Z_i)
\end{aligned}
$$

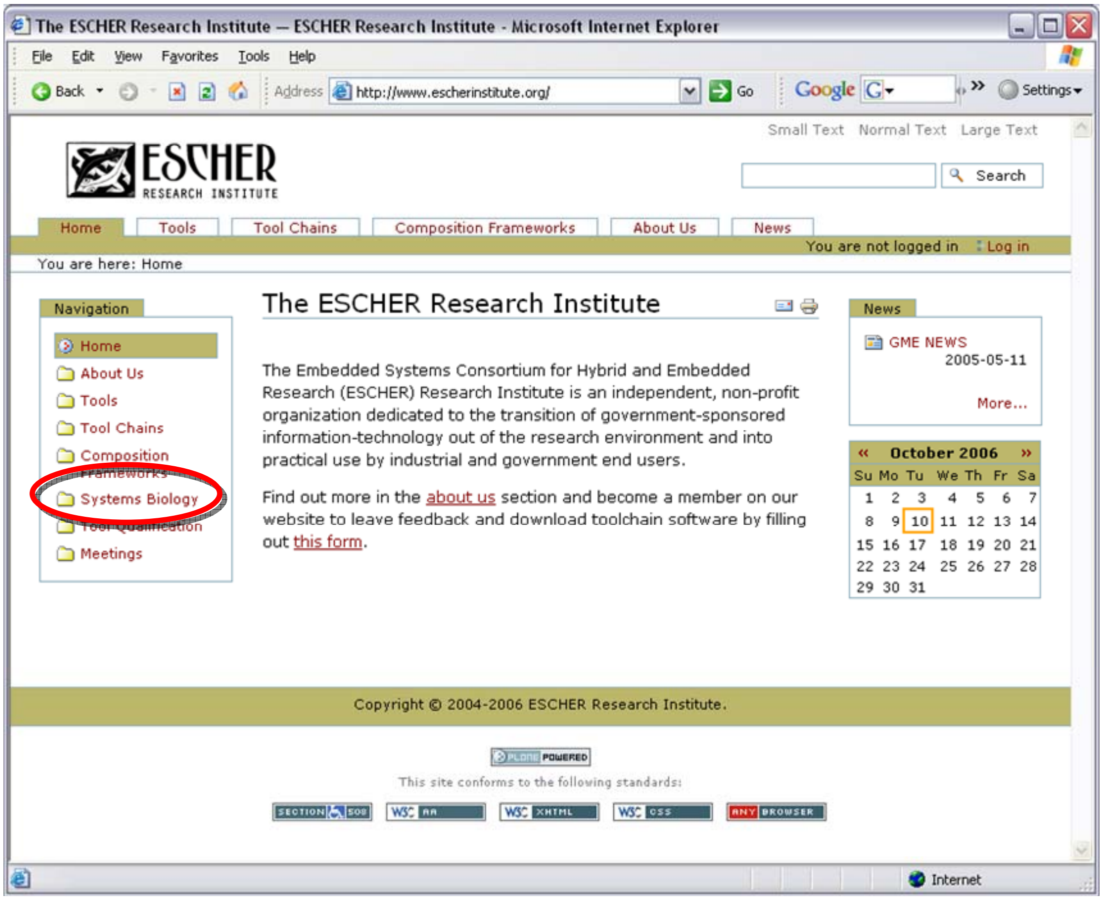
Variable	β-catenein	Goosecoid	Vox	Bozozok	Bmp2b	Chordino
mRNA		g	v	z		
Cellular Protein	c	G	V	Z	b	d
Secreted Protein					B	D

Figure 13: Zebra Fish Mathematical Model

A MATLAB program was developed to solve these equations. There were some difficulties due to the lack of information on which set/type of equations capture the interactions correctly. Simulation faced some challenges due to non-linear partial differential equations, large number of parameters, and lack of quantitative description in the literature. The search space for the model was pretty large due to the above referenced issues. A further complication was encountered, since the experimental outputs were available as photos which could not be compared quantitatively without sophisticated image processing and pattern matching. Human observation was required to establish the validity of the simulation.

3.3 ESCHER Quality Controlled Repository

The ESCHER Quality Controlled Repository was augmented with a Systems Biology segment. A collection of systems biology tools were hosted on this quality controlled repository. Figures 14-18 show snapshots of the repository. The repository can be reached at: http://www.escherinsitute.org.



The ESCHER Research Institute

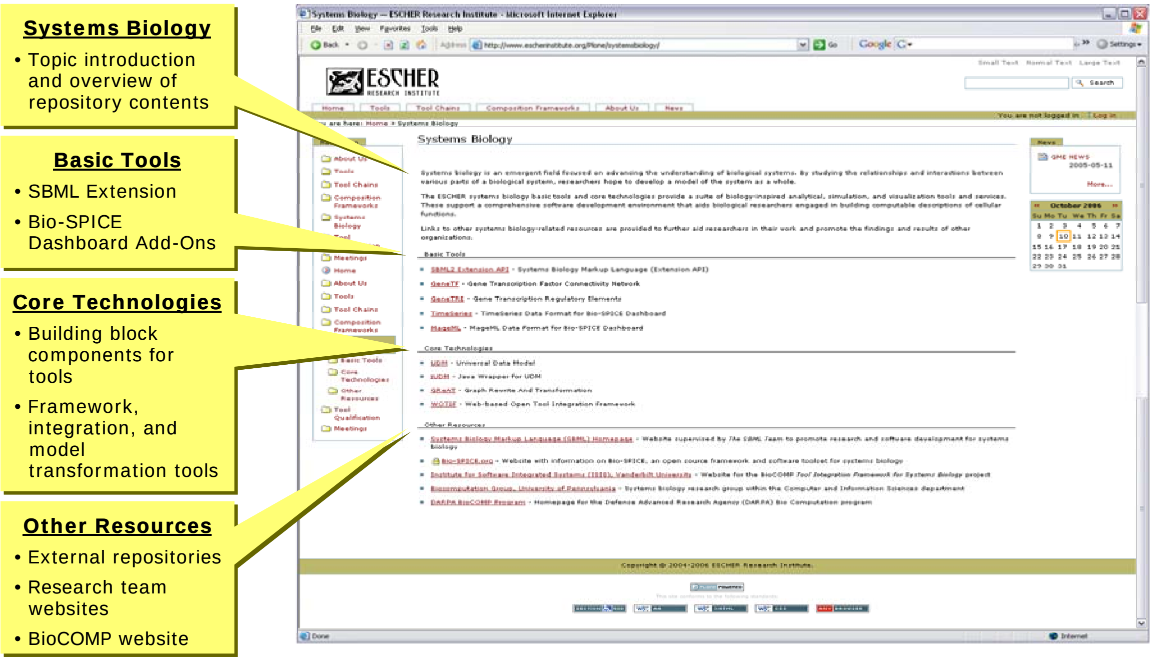
The Embedded Systems Consortium for Hybrid and Embedded Research (ESCHER) Research Institute is an independent, non-profit organization dedicated to the transition of government-sponsored information-technology out of the research environment and into practical use by industrial and government end users.

Find out more in the about us section and become a member on our website to leave feedback and download toolchain software by filling out this form.

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Copyright © 2004-2006 ESCHER Research Institute.

Figure 14: ESCHER Home Page



Systems Biology

- Topic introduction and overview of repository contents

Basic Tools

- SBML Extension
- Bio-SPICE Dashboard Add-Ons

Core Technologies

- Building block components for tools
- Framework, integration, and model transformation tools

Other Resources

- External repositories
- Research team websites
- BioCOMP website

Systems Biology — ESCHER Research Institute - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.escherinstitute.org/Plone/systemsbiology/
Go
Google
Small Text Normal Text Large Text
ESCHER RESEARCH INSTITUTE
Search
Home
Tools
Tool Chains
Composition Frameworks
About Us
News
You are not logged in
Log in
You are here: Home » Systems Biology
News
GME NEWS
2005-05-11
More...
October 2006
Done
Internet

Systems Biology

Systems biology is an emergent field focused on advancing the understanding of biological systems. By studying the relationships and interactions between various parts of a biological system, researchers hope to develop a model of the system as a whole.

The ESCHER systems biology basic tools and core technologies provide a suite of biology-inspired analytical, simulation, and visualization tools and services. These support a comprehensive software development environment that aids biological researchers engaged in building computable descriptions of cellular functions.

Links to other systems biology-related resources are provided to further aid researchers in their work and promote the findings and results of other organizations.

Basic Tools

- SBML2 Extension API - Systems Biology Markup Language (Extension API)
- GeneTF - Gene Transcription Factor Connectivity Network
- GeneTRE - Gene Transcription Regulatory Elements
- TimeSeries - TimeSeries Data Format for Bio-SPICE Dashboard
- MageML - MageML Data Format for Bio-SPICE Dashboard

Core Technologies

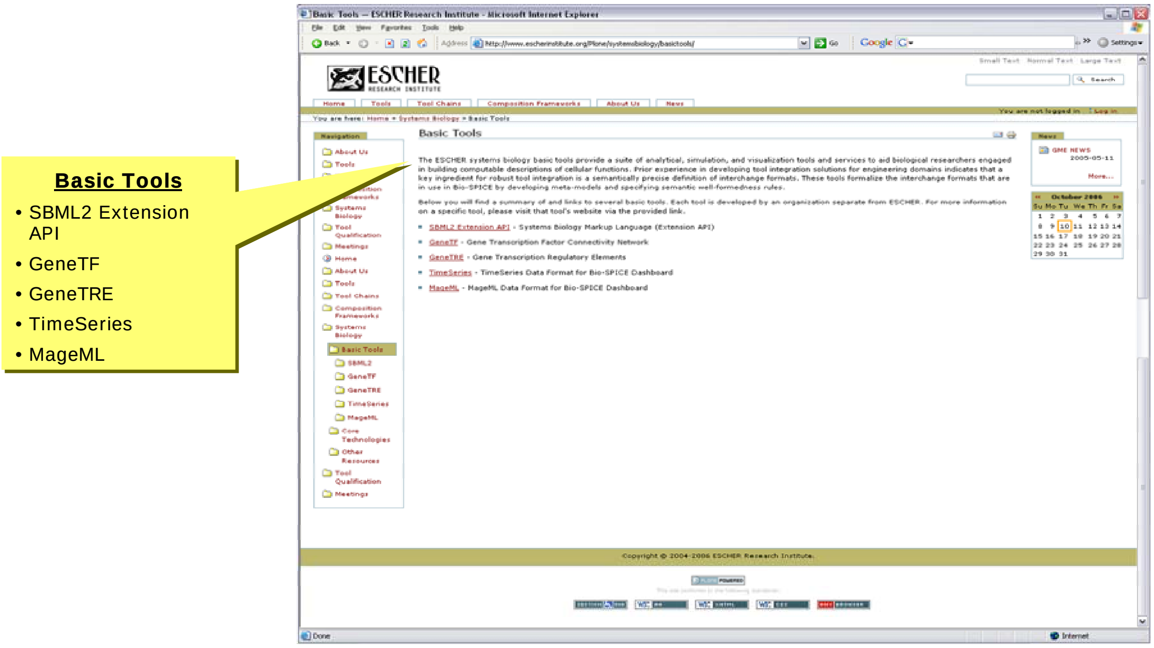
- UDM - Universal Data Model
- JUDM - Java Wrapper for UDM
- GReAT - Graph Rewrite And Transformation
- WOTIF - Web-based Open Tool Integration Framework

Other Resources

- Systems Biology Markup Language (SBML) Homepage - Website supervised by *The SBML Team* to promote research and software development for systems biology
- Bio-SPICE.org - Website with information on Bio-SPICE, an open source framework and software toolset for systems biology
- Institute for Software Integrated Systems (ISIS), Vanderbilt University - Website for the BioCOMP *Tool Integration Framework for Systems Biology* project
- Biocomputation Group, University of Pennsylvania - Systems biology research group within the Computer and Information Sciences department
- DARPA BioCOMP Program - Homepage for the Defence Advanced Research Agency (DARPA) Bio Computation program

Copyright © 2004-2006 ESCHER Research Institute.

Figure 15: ESCHER Systems Biology - Main Page


Basic Tools — ESCHER Research Institute - Microsoft Internet Explorer
http://www.escherinstitute.org/Plone/systemsbiology/basictools/
ESCHER RESEARCH INSTITUTE
Home
Tools
Tool Chains
Composition Frameworks
About Us
News
You are here: Home » Systems Biology » Basic Tools
Navigation
Basic Tools
SBML2
GeneTF
GeneTRE
TimeSeries
MageML
Copyright © 2004-2006 ESCHER Research Institute.

Basic Tools

- SBML2 Extension API
- GeneTF
- GeneTRE
- TimeSeries
- MageML

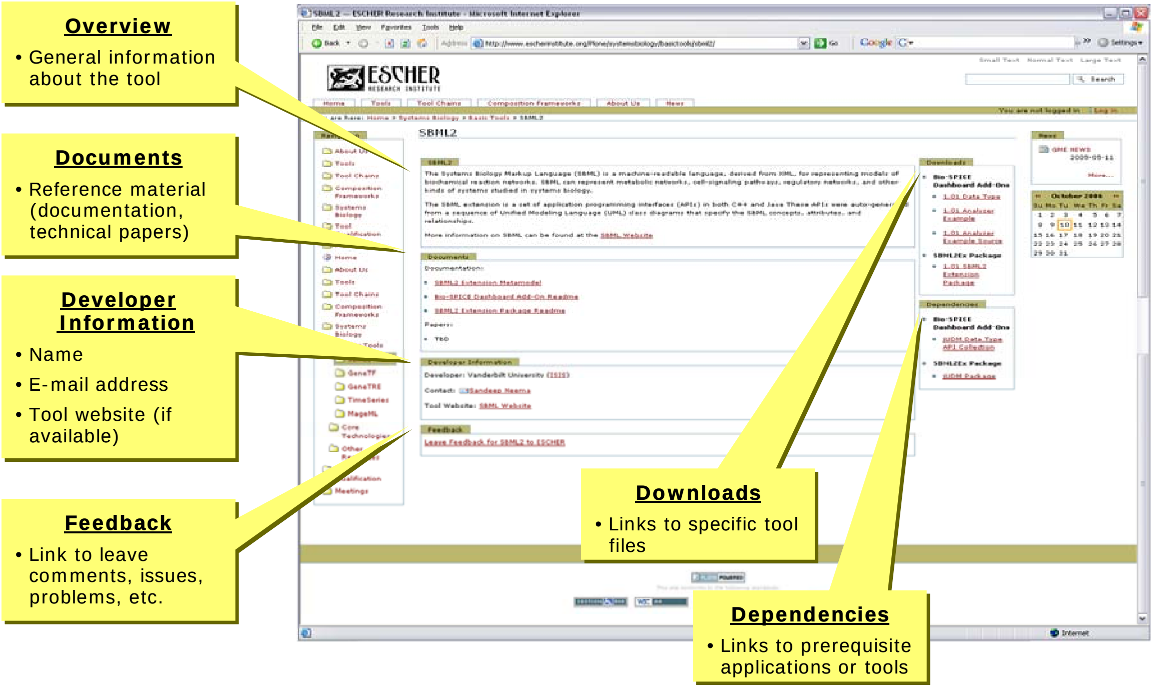
Basic Tools

The ESCHER systems biology basic tools provide a suite of analytical, simulation, and visualization tools and services to aid biological researchers engaged in building computable descriptions of cellular functions. Prior experience in developing tool integration solutions for engineering domains indicates that a key ingredient for robust tool integration is a semantically precise definition of interchange formats. These tools formalize the interchange formats that are in use in Bio-SPICE by developing meta-models and specifying semantic well-formedness rules.

Below you will find a summary of and links to several basic tools. Each tool is developed by an organization separate from ESCHER. For more information on a specific tool, please visit that tool's website via the provided link.

- SBML2 Extension API - Systems Biology Markup Language (Extension API)
- GeneTF - Gene Transcription Factor Connectivity Network
- GeneTRE - Gene Transcription Regulatory Elements
- TimeSeries - TimeSeries Data Format for Bio-SPICE Dashboard
- MageML - MageML Data Format for Bio-SPICE Dashboard

Figure 16: ESCHER Systems Biology - Basic Tools Page


SBML2 — ESCHER Research Institute - Microsoft Internet Explorer
http://www.escherinstitute.org/Plone/systemsbiology/basictools/sbml2/
ESCHER RESEARCH INSTITUTE
Home
Tools
Tool Chains
Composition Frameworks
About Us
News
You are here: Home » Systems Biology » Basic Tools » SBML2

Overview

- General information about the tool

Documents

- Reference material (documentation, technical papers)

Developer Information

- Name
- E-mail address
- Tool website (if available)

Feedback

- Link to leave comments, issues, problems, etc.

SBML2

SBML2

The Systems Biology Markup Language (SBML) is a machine-readable language, derived from XML, for representing models of biochemical reaction networks. SBML can represent metabolic networks, cell-signaling pathways, regulatory networks, and other kinds of systems studied in systems biology.

The SBML extension is a set of application programming interfaces (APIs) in both C++ and Java These APIs were auto-generated from a sequence of Unified Modeling Language (UML) class diagrams that specify the SBML concepts, attributes, and relationships.

More information on SBML can be found at the SBML Website

Documents

Documentation:

- SBML2 Extension Metamodel
- Bio-SPICE Dashboard Add-On Readme
- SBML2 Extension Package Readme

Papers:

- TBD

Developer Information

Developer: Vanderbilt University (ISIS)

Contact: Sandeep Neema

Tool Website: SBML Website

Feedback

Leave Feedback for SBML2 to ESCHER

Downloads

- Bio-SPICE Dashboard Add-Ons
  - 1.01 Data Type
  - 1.01 Analyzer Example
  - 1.01 Analyzer Example Source
- SBML2Ex Package
  - 1.01 SBML2 Extension Package

Dependencies

- Bio-SPICE Dashboard Add-Ons
  - UDM Data Type API Collection
- SBML2Ex Package
  - UDM Package

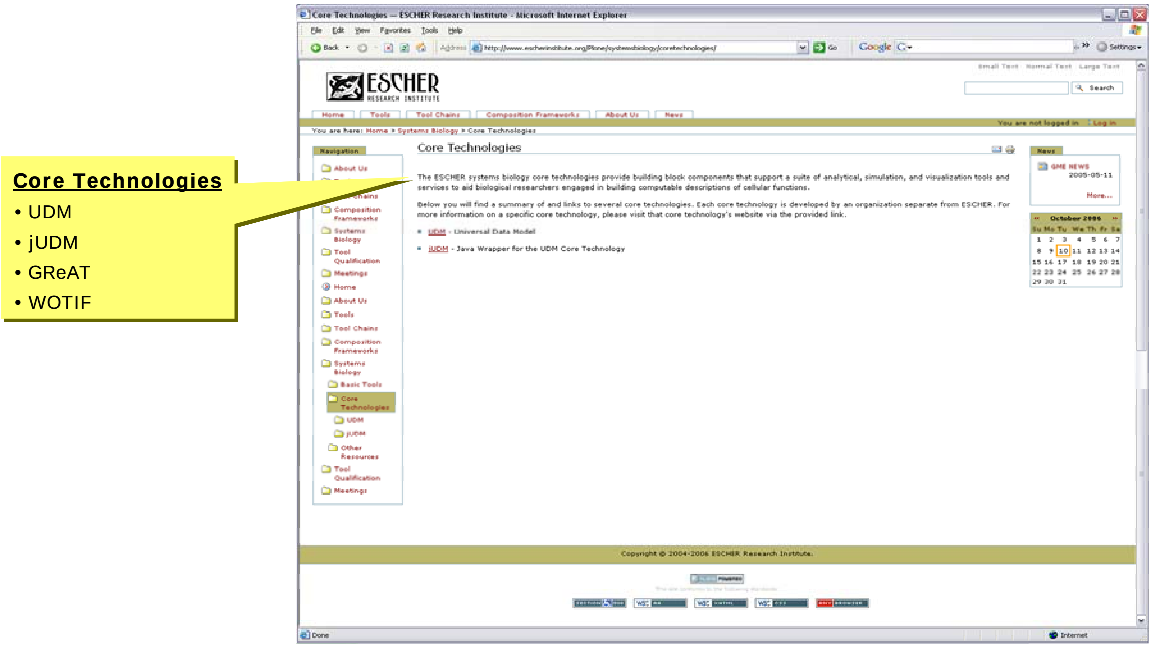
Downloads

- Links to specific tool files

Dependencies

- Links to prerequisite applications or tools

Figure 17: ESCHER Systems Biology - Basic Tools / SBML2


Core Technologies — ESCHER Research Institute - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://www.escherinstitute.org/Plone/systemsbiology/coretechnologies/
Go
Google
Settings
Small Text Normal Text Large Text
ESCHER RESEARCH INSTITUTE
Search
Home
Tools
Tool Chains
Composition Frameworks
About Us
News
You are not logged in
Log in
Navigation
News
GME NEWS 2005-05-11
More...
October 2006
Done
Internet

You are here: Home » Systems Biology » Core Technologies

Core Technologies

The ESCHER systems biology core technologies provide building block components that support a suite of analytical, simulation, and visualization tools and services to aid biological researchers engaged in building computable descriptions of cellular functions.

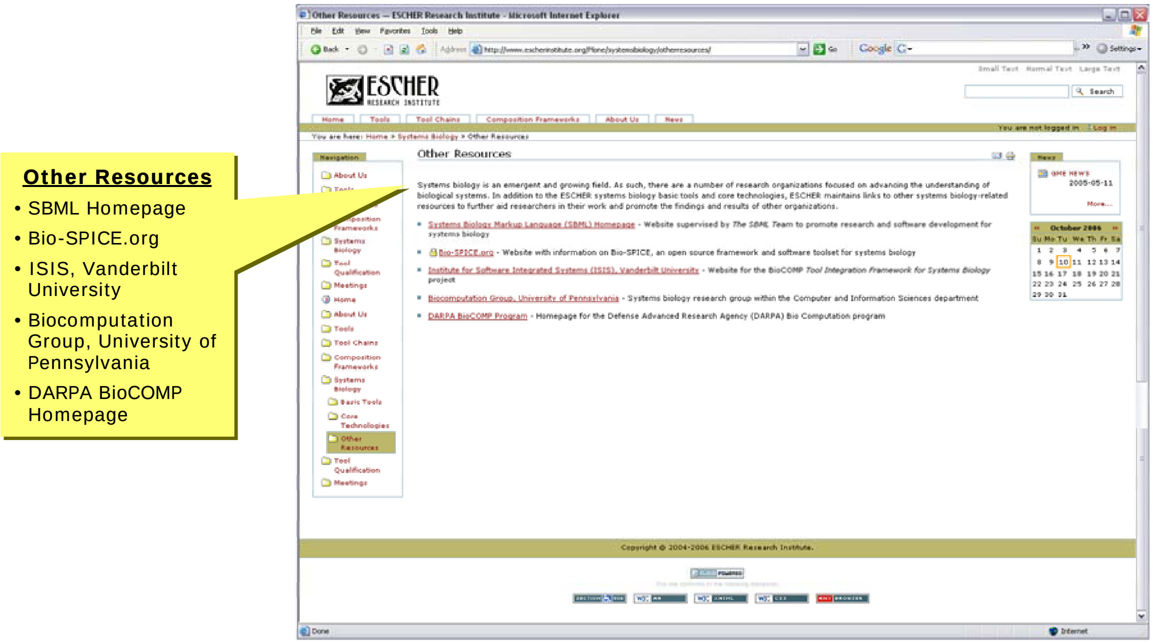
Below you will find a summary of and links to several core technologies. Each core technology is developed by an organization separate from ESCHER. For more information on a specific core technology, please visit that core technology's website via the provided link.

- UDM - Universal Data Model
- jUDM - Java Wrapper for the UDM Core Technology

Copyright © 2004-2006 ESCHER Research Institute.

Core Technologies
- UDM
- jUDM
- GReaT
- WOTIF

Figure 18: ESCHER Systems Biology - Core Technologies Page


Other Resources — ESCHER Research Institute - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://www.escherinstitute.org/Plone/systemsbiology/otherresources/
Go
Google
Settings
Small Text Normal Text Large Text
ESCHER RESEARCH INSTITUTE
Search
Home
Tools
Tool Chains
Composition Frameworks
About Us
News
You are not logged in
Log in
Navigation
News
GME NEWS 2005-05-11
More...
October 2006
Done
Internet

You are here: Home » Systems Biology » Other Resources

Other Resources

Systems biology is an emergent and growing field. As such, there are a number of research organizations focused on advancing the understanding of biological systems. In addition to the ESCHER systems biology basic tools and core technologies, ESCHER maintains links to other systems biology-related resources to further aid researchers in their work and promote the findings and results of other organizations.

- Systems Biology Markup Language (SBML) Homepage - Website supervised by *The SBML Team* to promote research and software development for systems biology
- Bio-SPICE.org - Website with information on Bio-SPICE, an open source framework and software toolset for systems biology
- Institute for Software Integrated Systems (ISIS), Vanderbilt University - Website for the BioCOMP *Tool Integration Framework for Systems Biology* project
- Biocomputation Group, University of Pennsylvania - Systems biology research group within the Computer and Information Sciences department
- DARPA BioCOMP Program - Homepage for the Defense Advanced Research Agency (DARPA) Bio Computation program

Copyright © 2004-2006 ESCHER Research Institute.

Other Resources
- SBML Homepage
- Bio-SPICE.org
- ISIS, Vanderbilt University
- Biocomputation Group, University of Pennsylvania
- DARPA BioCOMP Homepage

Figure 19: ESCHER Systems Biology - Other Resources Page

4. Summary

This project created formal and precisely defined interchange formats, leveraging standards based techniques, and tools developed in prior/related government sponsored programs. The interchange format metamodels were used to generate API-s that were utilized by other researchers in the program. The project demonstrated the use of an advanced tool integration framework such as OTIF that employs formal description of interchange formats, and performs semantic transformations to integrate tools. The project demonstrated integrating OTIF and Dashboard, and created heterogeneous workflows that could involve real-time interactions, and heterogeneous and distributed computing resources such as cluster computing. Finally, the project developed a technical repository based upon the ESCHER model of a quality-controlled repository, which provides services for archiving, quality guidelines, and quality assessment, bug and issues reporting and tracking for software tools. The results of the project were disseminated via the project website and the Dashboard Update Center, and are available through the ESCHER quality controlled repository.

References

[1] Karsai, G., Sztipanovits, J., Ledeczi, A. and Bapty, T. “Model-integrated development of embedded software.” In *Proceedings of the IEEE*, Volume 91, pages 145-164, 2003.

[2] Karsai, G., Lang, A., and S. Neema, S. “Design Patterns for Tool Integration,” *Journal of Software and Systems Modeling*, Volume 4, Number 2, pp 157-170.

[3] Vadigepalli, R., Chakravarthula, P., Zak, D.E., Schwaber, J.S., Gonye G.E. “PAINT: a promoter analysis and interaction network generation tool for gene regulatory network identification,” *OMICS*. 2003 Fall;7(3):235-52.

[4] Liao, J.C., et al. "Network component analysis: reconstruction of regulatory signals in biological systems,”.Proc Natl Acad Sci U S A. 2003 Dec 23;100 (26):15522-7.

[5] BioSPICE Dashboard Website – https://sourceforge.net/projects/biospice

[6] Eclipse – http://www.eclipse.org

[7] ESCHER Website – http://www.escherinstitute.org

[8] GME Website - http://www.isis.vanderbilt.edu/projects/gme/

[9] GReAT Website - http://www.escherinstitute.org/Plone/tools/suites/mic/great

[10] ISIS – http://www.isis.vanderbilt.edu

[11] Java NetBeans – http://java.sun.com/j2se/1.5.0/download-netbeans.html

[12] OTIF Website - http://www.escherinstitute.org/Plone/frameworks/otif

[13] PAINT Website - http://www.dbi.tju.edu/dbi/tools/paint/

[14] SBML – http://sbml.org

[15] UDM Website - http://www.escherinstitute.org/Plone/tools/suites/mic/udm

[16] UML Website - http://www.uml.org/

[17] OCL Specification - http://www.omg.org/technology/documents/formal/ocl.htm

[18] Gilardelli, C.N., Pozzoli, O., Sordino, P., Matassi, G. and Cotelli, F. “Functional and Hierarchical Interactions Among Zebrafish vox/vent Homeobox Genes,” *Developmental Dynamics*, 230:494-508, 2004.

Glossary

API	Application Programming Interface.
GReAT	Graph Rewriting and Transformations. A language and toolsuite for constructing model transformation programs.
GME	Generic Modeling Environment. A metaprogrammable visual model editor.
NCA	Network Connectivity Analysis tool that processes information captured in gene/transcription factor maps. Used in systems biology.
OTIF	Open Tool Integration Framework.
PAINT	A clustering analysis tool that produces gene/transcription factor maps. Used in systems biology.
SBML	Systems Biology Markup Language. An XML-based markup language for systems biology modeling.
UDM	Unified Data Model. A software package that generates C++ and Java API-s from UML metamodels, that could be used to access models stored in GME, in XML files, or as CORBA structures.
UML	Unified Modeling Language. A modeling language for modeling in object-oriented analysis and software design.
XML	Extensible Markup Language.
XSLT	XML Stylesheet Language for Translations. XML-based scripting language for describing simple transformations on XML data.